Growth rate and gene expression of Faecalibacterium prausnitzii relate to methane production

Boyang Zhang, Shili Lin, and Zhongtang Yu

This document contains all the **statistical analyses conducted for the manuscript. All** the **data** used **for the analyses can be found** in the **additional file or this link:** *https://github.com/boyangzhang1993/PTR_Methane_Bioinformatic*

Question 1: Does growth rate/relative abundance correlate with methane production?

1.1 Load data

```
ptrAbMethane = read.csv(file = 'ptrAbundanceGrouped.csv', header = T)
#str(ptrAbMethane)
# head(ptrAbMethane)
```

1.2 Wilcoxon test and FDR correction

```
pCollection = c()
for (i in c(4: ncol(ptrAbMethane))) {
  #i = 3
  wTest = wilcox.test(x = ptrAbMethane[,i][ptrAbMethane$Class == "high"],
              y = ptrAbMethane[,i][ptrAbMethane$Class == "low"],
              paired = F)

  pCollection = append(pCollection, wTest$p.value)
}
adjustedP = p.adjust(pCollection, method = "BH", n = length(pCollection))


testNameWS = colnames(ptrAbMethane)[c(4: ncol(ptrAbMethane))]
```

```
dfAll = data.frame("binID"=testNameWS)
taxasC = c()
meansC = c()
sdsC = c()
library(stringr)
for (i in c(1:length(dfAll$binID))) {
  # i = 35
  if (i <= 33) {
    bin = dfAll$binID[i]
  LMY_ptrs = ptrAbMethane[,i+3][ptrAbMethane$Class == "low"]
  HMY_ptrs = ptrAbMethane[,i+3][ptrAbMethane$Class == "high"]
  }else{
    bin = dfAll$binID[i]
    # f_s =  str_locate_all(pattern = "f", string = bin)[[1]][1,1]
    # bin = substring(text = dfAll$binID[i], first = 0,last = f_s-2)
  LMY_ptrs = ptrAbMethane[,i+3][ptrAbMethane$Class == "low"]*100
  HMY_ptrs = ptrAbMethane[,i+3][ptrAbMethane$Class == "high"]*100
  }

  meanLMY = round(mean(LMY_ptrs),3)
  sdLMY = round(sd(LMY_ptrs),3)
  meanHMY = round(mean(HMY_ptrs),3)
  sdHMY = round(sd(HMY_ptrs),3)
```

```
  meanHMY_LMY = paste(meanHMY, "(", sdHMY, ")",sep = "")
  meansC = append(meansC, meanHMY_LMY)
  sdHMY_LMY = paste(meanLMY, "(", sdLMY, ")",sep = "")
  sdsC = append(sdsC, sdHMY_LMY)

}
# length(taxasC)
dfAll$Taxa = taxasC
dfAll$HMY = meansC
dfAll$LMY = sdsC
dfAll$adjustedP = round(adjustedP, 3)

dfPTR = dfAll[c(1:33), ]

dfPTR = dfPTR[order(dfPTR$adjustedP), ]
dfAbundance = dfAll[c(34:nrow(dfAll)),]
dfAbundance = dfAbundance[order(dfAbundance$adjustedP),]
```

1.3 Results of growth rates

```
head(dfPTR)
```

```
##                                                                 binID          HMY
## 20 PTR_Genus_faecalibacterium_Species_faecalibacterium.prausnitzii  1.724(0.14)
## 2                                                PTR_Genus_aminipila  1.74(0.073)
## 6                      PTR_Genus_bacillus_Species_bacillus.pumilus 1.515(0.096)
## 17                                    PTR_Genus_corynebacterium 2.048(0.168)
## 30     PTR_Genus_ruminococcus_Species_ruminococcus.champanellensis 1.828(0.193)
## 7                                       PTR_Genus_bacteroides 1.733(0.112)
##             LMY adjustedP
## 20 1.525(0.102)     0.034
## 2  1.669(0.119)     0.176
## 6  1.415(0.071)     0.176
## 17 2.138(0.178)     0.214
## 30 1.679(0.162)     0.287
## 7  1.802(0.077)     0.337
```

1.4 Results of relative abundances

```
head(dfAbundance)
```

```
##                                                                                 binID
## 48                                                        Abundance_Genus_alistipes
## 54                                Abundance_Genus_anaerostipes_Species_anaerostipes.hadrus
## 62 Abundance_Genus_bacillus_Species_bacillus.coagulans_Species_bacillus.circulans
## 81                                    Abundance_Genus_blautia_Species_blautia.sp..sc05b48
## 86                               Abundance_Genus_bradyrhizobium_Species_bradyrhizobium.sp.
## 90                 Abundance_Genus_bradyrhizobium_Species_bradyrhizobium.sp..ccbau.051011
##              HMY          LMY adjustedP
```

```
## 48 3.201(0.995) 1.787(0.413)     0.011
## 54  0.058(0.07)         0(0)     0.011
## 62 0.009(0.006) 0.141(0.101)     0.011
## 81 0.001(0.001) 0.121(0.161)     0.011
## 86 0.077(0.128) 0.002(0.002)     0.011
## 90 0.118(0.056) 0.003(0.005)     0.011
```

1.5 Plot 1 A and B

```
ptrNames = c("PTR_Genus_faecalibacterium_Species_faecalibacterium.prausnitzii",
             "Abundance_Genus_faecalibacterium_Species_faecalibacterium.prausnitzii",
             "PTR_Genus_prevotella", "Abundance_Genus_prevotella")


ploty = c()
plotClass = c()
plotx = c()
ptrAbMethane[,"Abundance_Genus_faecalibacterium_Species_faecalibacterium.prausnitzii"] = ptrAbMethane[,
for (ptrName in ptrNames) {

  ploty = append(ploty, ptrAbMethane[,ptrName])
  plotClass = append(plotClass, ptrAbMethane$Class)

  plotx = append(plotx, rep(ptrName, 20))

}
# length(plotx)


plotDf = data.frame(x = plotx, y = ploty, class = plotClass)
plotDf =subset(plotDf, plotClass!= "intermediate")

data_summary <- function(data, varname, groupnames){
  require(plyr)
  summary_func <- function(x, col){
    c(mean = mean(x[[col]], na.rm=TRUE),
      sd = sd(x[[col]], na.rm=TRUE))
  }
  data_sum<-ddply(data, groupnames, .fun=summary_func,
                  varname)
  # data_sum <- rename(data_sum, c("mean" = varname))
 return(data_sum)
}

df3 <- data_summary(plotDf, varname="y",
                    groupnames=c("x","class"))
```

```
## Loading required package: plyr
```

```
df3$y = df3$mean
```

```
ptrNames = testNameWS[which(adjustedP < 0.1)]


ploty = c()
plotClass = c()
plotx = c()
for (ptrName in ptrNames) {

  ploty = append(ploty, ptrAbMethane[,ptrName])
  plotClass = append(plotClass, ptrAbMethane$Class)

  plotx = append(plotx, rep(ptrName, 20))

}


# Plot data
plotDf = data.frame(x = plotx, y = ploty, class = plotClass)
plotDf =subset(plotDf, plotClass!= "intermediate")
plotDf$adjustP = adjustedP[which(adjustedP < 0.1)]

plotDf2 = plotDf[order(plotDf$adjustP),]


# 1B
library(ggplot2)
library(gridtext)
plotDFfa1130 = subset(df3, x == "Abundance_Genus_prevotella" )
p1130Fa = ggplot(data=plotDFfa1130 , aes(x=x, y=y, fill=class))+
  geom_errorbar(aes(ymin=y-sd, ymax=y+sd), width=.2,
                position=position_dodge(.9)) +
   geom_bar(stat="identity", position=position_dodge(), aes(color = class)) +
  scale_fill_brewer(palette="Paired") +
  theme_minimal() + labs(y = "Relative abundance")+
  scale_x_discrete(labels = c(expression(italic("Prevotella"))))+
  scale_color_manual(values = c("tan3", "dodgerblue3"))+
  scale_fill_manual(values = c("tan1", "dodgerblue1"))
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```

```
plotDF1130 = subset(df3, x == "PTR_Genus_prevotella" )
p1130 = ggplot(data= plotDF1130, aes(x=x, y=y, fill=class)) +
  geom_errorbar(aes(ymin=y-sd, ymax=y+sd), width=.2,
                position=position_dodge(.9))+
   geom_bar(stat="identity", position=position_dodge(), aes(color = class))+
  scale_fill_brewer(palette="Paired") +
  scale_x_discrete(labels = c(expression(italic("Prevotella"))))+
  theme_minimal()+ labs(y = "Growth rate")+
  ylim(0, 2.5)+
  scale_color_manual(values = c("tan3", "dodgerblue3"))+
  scale_fill_manual(values = c("tan1", "dodgerblue1"))
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```

```
library(gridtext)
plotDFfa2756 = subset(df3, x == "Abundance_Genus_faecalibacterium_Species_faecalibacterium.prausnitzii"

p2756Fa = ggplot(data=plotDFfa2756 , aes(x=x, y=y, fill=class))+
  geom_errorbar(aes(ymin=y-sd, ymax=y+sd), width=.2,
                position=position_dodge(.9))+
   geom_bar(stat="identity", position=position_dodge(), aes(color = class)) +
  scale_fill_brewer(palette="Paired") +
  theme_minimal() + labs(y = "Relative abundance (%)")+
  scale_x_discrete(labels = c(expression(italic("F. prausnitzii"))))+
  scale_color_manual(values = c("tan3", "dodgerblue3"))+
  scale_fill_manual(values = c("tan1", "dodgerblue1"))+ theme(legend.position = "none")
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```

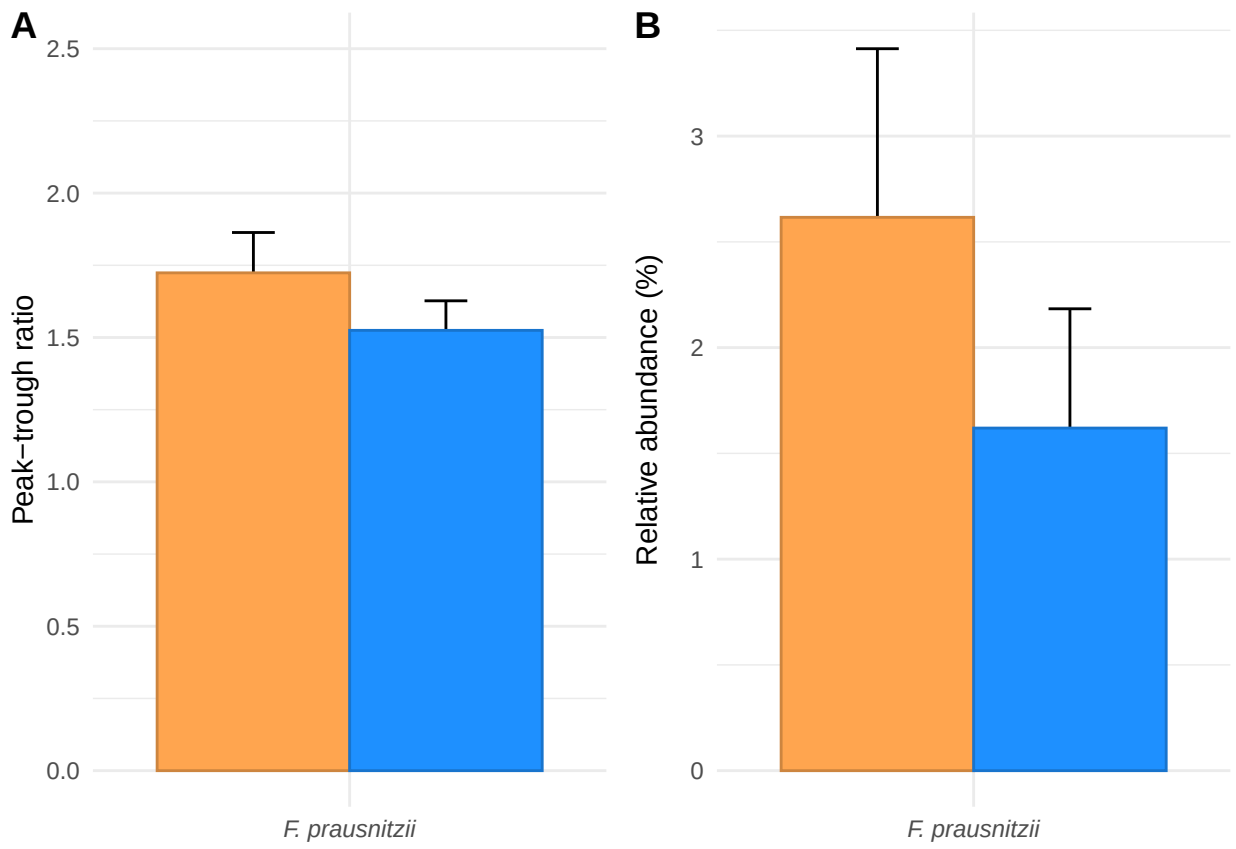
```
plotDF2756 = subset(df3, x == "PTR_Genus_faecalibacterium_Species_faecalibacterium.prausnitzii" )
p2756 = ggplot(data= plotDF2756, aes(x=x, y=y, fill=class)) +
  geom_errorbar(aes(ymin=y-sd, ymax=y+sd), width=.2,
                position=position_dodge(.9))+
   geom_bar(stat="identity", position=position_dodge(), aes(color = class))+
  scale_fill_brewer(palette="Paired") +
  scale_x_discrete(labels = c(expression(italic("F. prausnitzii"))))+
  theme_minimal()+ labs(y = "Peak-trough ratio")+
  ylim(0, 2.5)+
  scale_color_manual(values = c("tan3", "dodgerblue3"))+
  scale_fill_manual(values = c("tan1", "dodgerblue1"))+ theme(legend.position = "none")
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```

```
library(ggpubr)
```

```
##
## Attaching package: 'ggpubr'

## The following object is masked from 'package:plyr':
##
##     mutate
```

```
p5 = ggarrange(p2756+rremove("xlab"), p2756Fa+rremove("xlab"),
          common.legend = F,
          labels = c("A", "B","C","D"),
          ncol = 2, nrow = 1,
          align = "hv",
          legend = "none")


p5
```




```
# To save plot uncomment this part
width = 100
height = width*(4.5/4)
ggsave(p5, device = "png",filename = "ptr.png",width = width,units = "mm", dpi = 1000,
       height = height)
```

Question 2: Does gene expression of Faecalibacterium prausnitzii differ in the rumen microbiomes between the LMY and the HMY sheep?

2.1 IMG gene database

```
library(stringr)
RNA_fp_data = read.csv(file = 'RNA_methane_fp.csv', header = T)
RNA_fp_data = RNA_fp_data[,-1]

# head(RNA_fp_data)
```

Correlations / Wilcoxon test

```
colstart = 6
pCollection = c()
for (i in c(colstart: ncol(RNA_fp_data))) {
  #i = 115
  wTest = wilcox.test(x = RNA_fp_data[,i][RNA_fp_data$Class == "high"],
              y = RNA_fp_data[,i][RNA_fp_data$Class == "low"],
              paired = F)

  pCollection = append(pCollection, wTest$p.value)
}
```

```
alpha = 0.1
adjustedP = round(p.adjust(pCollection, method = "BH", n = length(pCollection)),3)
length(adjustedP)
```

```
## [1] 1575
```

```
length(colnames(RNA_fp_data)[c(colstart: ncol(RNA_fp_data))])
```

```
## [1] 1575
```

```
colnames(RNA_fp_data)[c(colstart: ncol(RNA_fp_data))][which(adjustedP <= alpha)]
```

```
##  [1] "X2815274858" "X2815275791" "X2815278666" "X2836648020" "X2836648021"
##  [6] "X2836649234" "X2836650584" "X2836651018" "X2836652010" "X2836652211"
```

```
ko = read.csv("finished.ko.txt",header = TRUE, sep = "\t")
cog = read.csv("finished.cog.txt",header = TRUE, sep = "\t")
ptrNames = colnames(RNA_fp_data)[c(colstart: ncol(RNA_fp_data))]


ptrNameSelected = c()
KOSelected = c()
k0_names = c()
```

```
check_ptr = c()
cog_id_c = c()
cog_name_c = c()
EC_names = c()

for (ptrName in ptrNames) {
  # ptrName = ptrNames[3]
  i= i +1
  abname = substring(ptrName, 2, nchar(ptrName))
  # abname
  check_ptr = append(check_ptr, abname)

  # idToAdd = which(ko$gene_oid == as.numeric(abname))
  # idToAdd
  if (abname %in% ko$gene_oid) {
    idToAdd = which(ko$gene_oid == abname)

    if (length(ko$ko_id[idToAdd]) > 1) {
      # print("k0")
      # print(abname)
      ko_to_add =paste(ko$ko_id[idToAdd],sep = "_",collapse = "_")
      ko_to_add
      # print(i)
      KOSelected = append(KOSelected, ko_to_add)
      ptrNameSelected = append(ptrNameSelected, abname)
    }
    else{
      KOSelected = append(KOSelected, ko$ko_id[idToAdd])
      ptrNameSelected = append(ptrNameSelected, abname)
    }

    # KOSelected = append(KOSelected, ko$ko_id[idToAdd])
    if (length(ko$ko_name[idToAdd]) > 1) {
      # print("k0")
      # print(abname)
      ko_to_add =paste(ko$ko_name[idToAdd],sep = "_",collapse = "_")
      k0_names = append(k0_names, ko_to_add)
    }
    else{
      k0_names = append(k0_names, ko$ko_name[idToAdd])

    }
     if (length(ko$EC[idToAdd]) > 1) {
      # print("k0")
      # print(abname)
      ko_to_add =paste(ko$EC[idToAdd],sep = "_",collapse = "_")
      EC_names = append(EC_names, ko_to_add)
    }
    else{
      EC_names = append(EC_names, ko$EC[idToAdd])

    }
    if (length(EC_names) != length(k0_names)) {
```

```
      print(i)
      stop()
    }



  }else{
    # print(paste("Did not in KO:", abname))
    ptrNameSelected = append(ptrNameSelected, abname)
    KOSelected = append(KOSelected, "-")
    k0_names = append(k0_names, "-")
    EC_names = append(EC_names, "-")
  }
  if (length(ptrNameSelected) != length(KOSelected)) {
    print(i)
    stop()
  }
  if (abname %in% cog$gene_oid) {
    idToAdd_COG = which(cog$gene_oid == abname)

    # cog_id_to_add

    # cog
    # library("jsonlite")
    # cog_link = paste('https://www.ncbi.nlm.nih.gov/research/cog/api/cog/?cog=', cog_id_to_add, '&form
    # cog_json <- jsonlite::fromJSON(cog_link)
    # cog_ncbi = cog_json[["results"]][["cog"]][["funcats"]][[1]]
    # cog_cate = cog_ncbi$name
    if (length(idToAdd_COG) >1 ) {
      cog_id_to_add = cog[idToAdd_COG,]$cog_id

      cog_cate_collapse = paste(cog_id_to_add,sep = "",collapse = "_")
      cog_id_c = append(cog_id_c, cog_cate_collapse)
      cog_id_to_add = cog[idToAdd_COG,]$cog_name

      cog_cate_collapse = paste(cog_id_to_add,sep = "",collapse = "_")
      cog_name_c = append(cog_name_c, cog_cate_collapse)

      # cog_category_c = append(cog_category_c, cog_cate_collapse)
    }else{
      cog_id_to_add = cog[idToAdd_COG,]$cog_id

      cog_id_c = append(cog_id_c, cog_id_to_add)

      cog_id_to_add = cog[idToAdd_COG,]$cog_name

      cog_name_c = append(cog_name_c, cog_id_to_add)
    }


  }else{
    cog_id_c = append(cog_id_c,  "-")
cog_name_c = append(cog_name_c, "-")
```

```
    }



}

# length(ptrNameSelected)
# length(kO_names)
# length(KOSelected)
# length(names(RNA_fp_data)[colstart:ncol(RNA_fp_data)])
```

```
names(RNA_fp_data)[colstart:ncol(RNA_fp_data)] = ptrNameSelected
ptrNames = colnames(RNA_fp_data)[c(colstart: ncol(RNA_fp_data))]
# dfAll = data.frame("GO"=ptrNameSelected, "KO" = KOSelected, "KO_name" = kO_names, "COG category" = co
dfAll = data.frame("IMG"=ptrNameSelected, "KO" = KOSelected, "KO_name" = kO_names,"EC" = EC_names,"COG

# RNA_fp_data
# taxas = read.csv(file = "taxa.csv",header = T)
# dfAll$Name[1347]
# taxasC = c()
meansC = c()
sdsC = c()
log2_CPM_c = c()
library(stringr)
for (i in c(1:length(dfAll$IMG))) {
  # print(i+4)
  # i = 1
  LMY_ptrs = RNA_fp_data[,i+5][RNA_fp_data$Class == "low"]
  HMY_ptrs = RNA_fp_data[,i+5][RNA_fp_data$Class == "high"]

  meanLMY = round(mean(LMY_ptrs),3)
  sdLMY = round(sd(LMY_ptrs),3)
  meanHMY = round(mean(HMY_ptrs),3)
  sdHMY = round(sd(HMY_ptrs),3)
  log2_CPM = round(log2(meanLMY/meanHMY),3)
  if (!is.na(log2_CPM) & abs(log2_CPM) != Inf) {
    log2_CPM_c = append(log2_CPM_c, log2_CPM)
  }else{
    log2_CPM_c = append(log2_CPM_c, NA)
  }


  meanHMY_LMY = paste(meanHMY, "(", sdHMY, ")",sep = "")
  meansC = append(meansC, meanHMY_LMY)
  sdHMY_LMY = paste(meanLMY, "(", sdLMY, ")",sep = "")
  sdsC = append(sdsC, sdHMY_LMY)

}
# length(dfAll$KO)
length(meansC)
```

```
## [1] 1575
```

```
# dfAll$Taxa = taxasC
dfAll$HMY = meansC
dfAll$LMY = sdsC
dfAll$adjustedP = round(adjustedP, 3)
dfAll$LOGFC = log2_CPM_c

# write.csv(dfAll[with(dfAll, order(adjustedP, -LOGFC)), ], "df_cog_kegg.csv")
# dfAll
# head(dfAll)
```

2.2 KEGG database

```
RNA_fp_data = read.csv(file = 'RNA_methane_fp.csv', header = T)
RNA_fp_data = RNA_fp_data[,-1]

RNA_fp_data_ko = RNA_fp_data
ko = read.csv("finished.ko.txt",header = TRUE, sep = "\t")

ptrNames = colnames(RNA_fp_data_ko)[c(colstart: ncol(RNA_fp_data))]
ptrNameSelected = c()
KOSelected = c()
k0_names = c()
for (ptrName in ptrNames) {
  # ptrName = ptrNames[1]
  abname = substring(ptrName, 2, nchar(ptrName))
  if (abname %in% ko$gene_oid) {
    idToAdd = which(ko$gene_oid == as.numeric(abname))
    ptrNameSelected = append(ptrNameSelected, abname)

    if (length(ko$ko_id[idToAdd]) > 1) {
      # print("k0")
      # print(abname)
      KOSelected = append(KOSelected, ko$ko_id[idToAdd][1])
    }
    else{
      KOSelected = append(KOSelected, ko$ko_id[idToAdd])

    }

    # KOSelected = append(KOSelected, ko$ko_id[idToAdd])
    if (length(ko$ko_name[idToAdd]) > 1) {
      # print("k0")
      # print(abname)
      k0_names = append(k0_names, ko$ko_name[idToAdd][1])
    }
    else{
      k0_names = append(k0_names, ko$ko_name[idToAdd])

    }
```

```
  }else{
    # print(paste("Did not in KO:", abname))
    ptrNameSelected = append(ptrNameSelected, abname)
    KOSelected = append(KOSelected, "-")
    k0_names = append(k0_names, "-")
  }
}

# length(k0_names)
# length(KOSelected)
colnames(RNA_fp_data_ko)[c(colstart: ncol(RNA_fp_data))] = k0_names

new_ko = RNA_fp_data_ko[,10]


for (k in unique(k0_names)) {
  # k = unique(k0_names)[1]
  to_sum = which(colnames(RNA_fp_data_ko) == k)
  if (length(to_sum) >= 2) {
    RNA_fp_data_ko[, to_sum]
    to_add = rowSums(RNA_fp_data_ko[, to_sum])
  }else{
    to_add = RNA_fp_data_ko[, to_sum]
  }
  new_ko = cbind(new_ko, to_add)


}

# length(unique(k0_names))
# ncol(new_ko)
new_ko = new_ko[,-1]

colnames(new_ko) = unique(k0_names)

new_ko = as.data.frame(new_ko)
new_ko_all = new_ko[,-4]
new_ko_all_methane = cbind(RNA_fp_data[,5],new_ko_all)
new_ko_all_methane = as.data.frame(new_ko_all_methane)
names(new_ko_all_methane)[1] = "Class"

colstart = 2
pCollection = c()
for (i in c(colstart: ncol(new_ko_all_methane))) {
  #i = 115
  wTest = wilcox.test(x = new_ko_all_methane[,i][new_ko_all_methane$Class == "high"],
              y = new_ko_all_methane[,i][new_ko_all_methane$Class == "low"],
              paired = F)

  pCollection = append(pCollection, wTest$p.value)
}
alpha = 0.1
adjustedP = round(p.adjust(pCollection, method = "BH", n = length(pCollection)),3)
```

```
df_kegg = data.frame("KO_name" = colnames(new_ko_all_methane)[-1])

meansC = c()
sdsC = c()
log2_CPM_c = c()
library(stringr)
for (i in c(1:length(df_kegg$KO_name))) {
  # print(i+4)
  # i = 1
  LMY_ptrs = new_ko_all_methane[,i+1][RNA_fp_data$Class == "low"]
  HMY_ptrs = new_ko_all_methane[,i+1][RNA_fp_data$Class == "high"]

  meanLMY = round(mean(LMY_ptrs),3)
  sdLMY = round(sd(LMY_ptrs),3)
  meanHMY = round(mean(HMY_ptrs),3)
  sdHMY = round(sd(HMY_ptrs),3)
  log2_CPM = round(log2(meanLMY/meanHMY),3)
  if (!is.na(log2_CPM) & abs(log2_CPM) != Inf) {
    log2_CPM_c = append(log2_CPM_c, log2_CPM)
  }else{
    log2_CPM_c = append(log2_CPM_c, NA)
  }


  meanHMY_LMY = paste(meanHMY, "(", sdHMY, ")",sep = "")
  meansC = append(meansC, meanHMY_LMY)
  sdHMY_LMY = paste(meanLMY, "(", sdLMY, ")",sep = "")
  sdsC = append(sdsC, sdHMY_LMY)

}
# length(df_kegg$KO)
length(meansC)
```

```
## [1] 372
```

```
df_kegg$HMY = meansC
df_kegg$LMY = sdsC
df_kegg$adjustedP = round(adjustedP, 3)
df_kegg$LOGFC = log2_CPM_c
df_kegg = df_kegg[order(df_kegg$adjustedP), ]
# df_kegg
# write.csv(df_kegg, "kegg.csv")
# new_ko_all_methane
# write.csv(new_ko_all_methane, "kegg_2.csv")
# head(new_ko_all_methane)
```

2.3 Plot 2

```
Pathway_butyrate = read.csv(file = 'cpm_rna.csv', header = T)
```

```
ptrNames = colnames(Pathway_butyrate)[-1]
ploty = c()
plotClass = c()
plotx = c()
for (ptrName in ptrNames) {
  # ptrName = ptrNames[1]

  ploty = append(ploty, Pathway_butyrate[,ptrName])
  plotClass = append(plotClass, Pathway_butyrate$Class)
  abname = ptrName

  plotx = append(plotx, rep(ptrName, 20))


}
length(plotx)
```

```
## [1] 120
```

```
plotDf = data.frame(x = plotx, y = ploty, class = plotClass)
plotDf =subset(plotDf, plotClass!= "intermediate")
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
data_summary <- function(data, varname, groupnames){
  require(plyr)
  # data = plotDf
  summary_func <- function(x, col){
    c(meanT = mean(x[[col]], na.rm=TRUE),
      sd = sd(x[[col]], na.rm=TRUE))
  }
  data_sum<-ddply(data, groupnames, .fun=summary_func,
                  varname)
  names(data_sum)
  # data_sum <- rename(data_sum, c("meanT" = varname))
 return(data_sum)
}
```

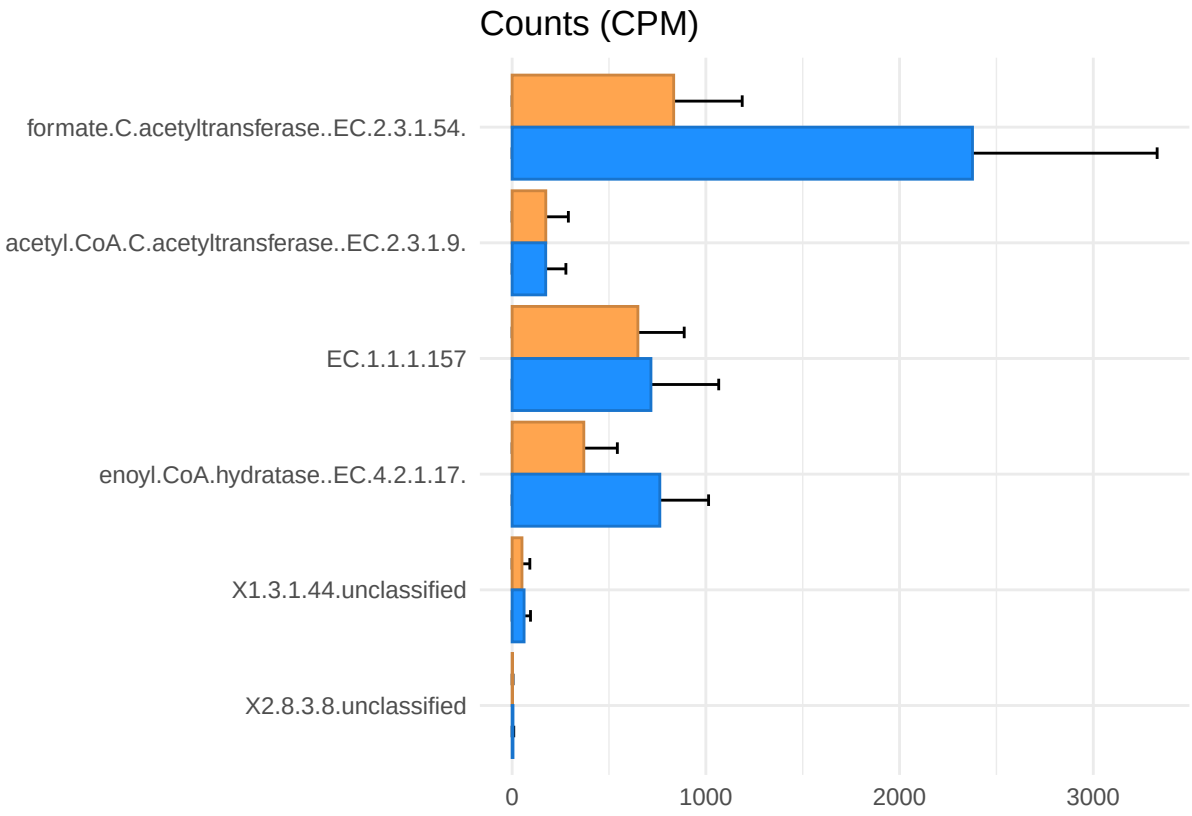
```
df3 <- data_summary(plotDf, varname="y",
                    groupnames=c("x","class"))
```

```
plotDf = as.data.frame(plotDf)
plotDf$y = as.numeric(plotDf$y)
df3 <- data_summary(plotDf, varname="y",
                    groupnames=c("x","class"))



df3$x <- factor(df3$x, levels=c("formate.C.acetyltransferase..EC.2.3.1.54.",
                                "acetyl.CoA.C.acetyltransferase..EC.2.3.1.9.",
                                "EC.1.1.1.157",
                                "enoyl.CoA.hydratase..EC.4.2.1.17.",
                                "ec.1.3.1.44",
                                "ec.2.8.3.8",
                                "X1.3.1.44.unclassified",
                                "X2.8.3.8.unclassified"))
```

```
library(forcats)
library(ggplot2)
library(ggthemes)

p5 = ggplot(df3, aes(x=fct_rev(x), y=meanT, fill=fct_rev(class))) +
    geom_errorbar(aes(ymin=0.1, ymax=meanT+sd), width=.2,
                  position=position_dodge(.9)) +
   geom_bar(stat="identity", position=position_dodge(), aes(color = fct_rev(class)))+
  coord_flip() +
  scale_fill_viridis_d(breaks = rev, direction = -1)+ theme_minimal()+xlab(" ")+ylab("")+
  scale_fill_manual(values = c("dodgerblue1", "tan1"))+
  scale_color_manual(values = c("dodgerblue3", "tan3"))+ggtitle("Counts (CPM)")+
  theme(legend.position = "none")
```

```
## Scale for 'fill' is already present. Adding another scale for 'fill', which
## will replace the existing scale.
```

```
p5
```




```
# height = 225
# width = 150
# height = width*(5/4)
# ggsave(p5, device = "png",filename = "pathwayB.png",width = width,units = "mm", dpi = 500,
#        height = height)
```

Question 3 : Do important gene expresssions selected differ among WRS, T and ANOVA test?

To increase the robustness of statistic tests, We used Top K list to integrate of results obtained from WRS, T and ANOVA test.

Functions for generate WRS, T, ANOVA test

```
# 1. IMG

df_IMG_tested = function(rna, wrs_test = F, t_test = F, anova_test = F){
  ## Remove first column
  RNA_fp_data = RNA_fp_data[,-1]
  # anova_test = T
  # wrs_test = F
  # t_test = F
  # Remove intermediate and 0 sums
  if (wrs_test | t_test) {
    RNA_fp_data = RNA_fp_data[-which(RNA_fp_data$Class=="intermediate"),]
    ko_names_selected_0 = c(6:ncol(RNA_fp_data))[as.numeric(which(colSums(RNA_fp_data[,c(6:ncol(RNA_fp_
    RNA_fp_data = RNA_fp_data[,-ko_names_selected_0]
  }


  RNA_fp_data_ko = RNA_fp_data
  colstart = 6

  pCollection = c()
  for (i in c(colstart: ncol(RNA_fp_data))) {
    # i = 6
    if (anova_test) {
      # paste(colnames(RNA_fp_data_ko)[i]," ~ ", "Class", sep = "")
      colnames(RNA_fp_data_ko)[i] = paste("X",i,sep = "")
      test_name = colnames(RNA_fp_data_ko)[i]
      anove_test = aov(formula = as.formula(paste(test_name,"~ ", "Class", sep = "")),data = RNA_fp_data
      s_anova = summary(anove_test)
      p_tested = s_anova[[1]][["Pr(>F)"]][1]
    }
    if (wrs_test) {
      wTest = wilcox.test(x = RNA_fp_data[,i][RNA_fp_data$Class == "high"],
                          y = RNA_fp_data[,i][RNA_fp_data$Class == "low"],
                          paired = F)
      p_tested = wTest$p.value
    }
    if (t_test) {
      tTest = t.test(x = RNA_fp_data[,i][RNA_fp_data$Class == "high"],
                     y = RNA_fp_data[,i][RNA_fp_data$Class == "low"],
                     paired = F,
                     var.equal = F)
      p_tested = tTest$p.value
    }
```

```
  # Box plot
  # putative aldouronate transport system substrate-binding protein
  # boxplot(RNA_fp_data$`putative aldouronate transport system substrate-binding protein`~RNA_fp_data

  pCollection = append(pCollection, p_tested)
}
ko = read.csv("finished.ko.txt",header = TRUE, sep = "\t")
cog = read.csv("finished.cog.txt",header = TRUE, sep = "\t")
ptrNames = colnames(RNA_fp_data)[c(colstart: ncol(RNA_fp_data))]
adjustedP = round(p.adjust(pCollection, method = "BH", n = length(pCollection)),3)

ptrNameSelected = c()
KOSelected = c()
k0_names = c()
check_ptr = c()
cog_id_c = c()
cog_name_c = c()
EC_names = c()

for (ptrName in ptrNames) {
  # ptrName = ptrNames[3]
  i= i +1
  abname = substring(ptrName, 2, nchar(ptrName))
  # abname
  check_ptr = append(check_ptr, abname)

  # idToAdd = which(ko$gene_oid == as.numeric(abname))
  # idToAdd
  if (abname %in% ko$gene_oid) {
    idToAdd = which(ko$gene_oid == abname)

    if (length(ko$ko_id[idToAdd]) > 1) {
      # print("k0")
      # print(abname)
      ko_to_add =paste(ko$ko_id[idToAdd],sep = "_",collapse = "_")
      ko_to_add
      # print(i)
      KOSelected = append(KOSelected, ko_to_add)
      ptrNameSelected = append(ptrNameSelected, abname)
    }
    else{
      KOSelected = append(KOSelected, ko$ko_id[idToAdd])
      ptrNameSelected = append(ptrNameSelected, abname)
    }

    # KOSelected = append(KOSelected, ko$ko_id[idToAdd])
    if (length(ko$ko_name[idToAdd]) > 1) {
      # print("k0")
      # print(abname)
      ko_to_add =paste(ko$ko_name[idToAdd],sep = "_",collapse = "_")
      k0_names = append(k0_names, ko_to_add)
    }
    else{
```

```
      k0_names = append(k0_names, ko$ko_name[idToAdd])

    }
    if (length(ko$EC[idToAdd]) > 1) {
      # print("k0")
      # print(abname)
      ko_to_add =paste(ko$EC[idToAdd],sep = "_",collapse = "_")
      EC_names = append(EC_names, ko_to_add)
    }
    else{
      EC_names = append(EC_names, ko$EC[idToAdd])

    }
    if (length(EC_names) != length(k0_names)) {
      print(i)
      stop()
    }



  }else{
    # print(paste("Did not in KO:", abname))
    ptrNameSelected = append(ptrNameSelected, abname)
    KOSelected = append(KOSelected, "-")
    k0_names = append(k0_names, "-")
    EC_names = append(EC_names, "-")
  }
  if (length(ptrNameSelected) != length(KOSelected)) {
    print(i)
    stop()
  }
  if (abname %in% cog$gene_oid) {
    idToAdd_COG = which(cog$gene_oid == abname)

    # cog_id_to_add

    # cog
    # library("jsonlite")
    # cog_link = paste('https://www.ncbi.nlm.nih.gov/research/cog/api/cog/?cog=', cog_id_to_add, '&fo
    # cog_json <- jsonlite::fromJSON(cog_link)
    # cog_ncbi = cog_json[["results"]][["cog"]][["funcats"]][[1]]
    # cog_cate = cog_ncbi$name
    if (length(idToAdd_COG) >1 ) {
      cog_id_to_add = cog[idToAdd_COG,]$cog_id

      cog_cate_collapse = paste(cog_id_to_add,sep = "",collapse = "_")
      cog_id_c = append(cog_id_c, cog_cate_collapse)
      cog_id_to_add = cog[idToAdd_COG,]$cog_name

      cog_cate_collapse = paste(cog_id_to_add,sep = "",collapse = "_")
      cog_name_c = append(cog_name_c, cog_cate_collapse)

      # cog_category_c = append(cog_category_c, cog_cate_collapse)
```

```
    }else{
      cog_id_to_add = cog[idToAdd_COG,]$cog_id

      cog_id_c = append(cog_id_c, cog_id_to_add)

      cog_id_to_add = cog[idToAdd_COG,]$cog_name

      cog_name_c = append(cog_name_c, cog_id_to_add)
    }


  }else{
    cog_id_c = append(cog_id_c,  "-")
    cog_name_c = append(cog_name_c, "-")
  }




}
names(RNA_fp_data)[colstart:ncol(RNA_fp_data)] = ptrNameSelected
ptrNames = colnames(RNA_fp_data)[c(colstart: ncol(RNA_fp_data))]
# dfAll = data.frame("GO"=ptrNameSelected, "KO" = KOSelected, "KO_name" = kO_names, "COG category" =
dfAll = data.frame("IMG"=ptrNameSelected, "KO" = KOSelected, "KO_name" = kO_names,"EC" = EC_names,"CO

# RNA_fp_data
# taxas = read.csv(file = "taxa.csv",header = T)
# dfAll$Name[1347]
# taxasC = c()
meansC = c()
sdsC = c()
log2_CPM_c = c()
library(stringr)
for (i in c(1:length(dfAll$IMG))) {
  # print(i+4)
  # i = 1
  LMY_ptrs = RNA_fp_data[,i+5][RNA_fp_data$Class == "low"]
  HMY_ptrs = RNA_fp_data[,i+5][RNA_fp_data$Class == "high"]

  meanLMY = round(mean(LMY_ptrs),3)
  sdLMY = round(sd(LMY_ptrs),3)
  meanHMY = round(mean(HMY_ptrs),3)
  sdHMY = round(sd(HMY_ptrs),3)
  log2_CPM = round(log2((meanLMY+1)/(meanHMY+1)),3)
  if (!is.na(log2_CPM) & abs(log2_CPM) != Inf) {
    log2_CPM_c = append(log2_CPM_c, log2_CPM)
  }else{
    log2_CPM_c = append(log2_CPM_c, NA)
  }


  meanHMY_LMY = paste(meanHMY, "(", sdHMY, ")",sep = "")
  meansC = append(meansC, meanHMY_LMY)
```

```
    sdHMY_LMY = paste(meanLMY, "(", sdLMY, ")",sep = "")
    sdsC = append(sdsC, sdHMY_LMY)

  }
  # length(dfAll$KO)
  length(meansC)

  # dfAll$Taxa = taxasC
  dfAll$HMY = meansC
  dfAll$LMY = sdsC
  dfAll$adjustedP = round(adjustedP, 3)
  dfAll$LOGFC = log2_CPM_c




  return(dfAll)
}

# 2. KEGG

df_kegg_tested = function(rna, wrs_test = F, t_test = F, anova_test = F){
  ## Remove first column
  RNA_fp_data = RNA_fp_data[,-1]

  # Remove intermediate and 0 sums
  if (wrs_test | t_test) {
    RNA_fp_data = RNA_fp_data[-which(RNA_fp_data$Class=="intermediate"),]
    ko_names_selected_0 = c(6:ncol(RNA_fp_data))[as.numeric(which(colSums(RNA_fp_data[,c(6:ncol(RNA_fp_
    RNA_fp_data = RNA_fp_data[,-ko_names_selected_0]
  }

  # 1. Rename gene names by KO names because we interested in expressions in KO
  RNA_fp_data_ko = RNA_fp_data
  ## Read KO data from JGI
  ko = read.csv("finished.ko.txt",header = TRUE, sep = "\t")
  ## colstart is the start index of gene
  colstart = 6
  ## Map gene names to KO names
  ### Recod all gene names
  ptrNames = colnames(RNA_fp_data_ko)[c(colstart: ncol(RNA_fp_data))]
  ptrNameSelected = c()
  KOSelected = c()
  k0_names = c()

  for (ptrName in ptrNames) {
    # ptrName = ptrNames[1]
    abname = substring(ptrName, 2, nchar(ptrName))
    if (abname %in% ko$gene_oid) {
      idToAdd = which(ko$gene_oid == as.numeric(abname))
      ptrNameSelected = append(ptrNameSelected, abname)

      if (length(ko$ko_id[idToAdd]) > 1) {
```

```
      # print("k0")
      # print(abname)
      KOSelected = append(KOSelected, ko$ko_id[idToAdd][1])
    }
    else{
      KOSelected = append(KOSelected, ko$ko_id[idToAdd])

    }

    # KOSelected = append(KOSelected, ko$ko_id[idToAdd])
    if (length(ko$ko_name[idToAdd]) > 1) {
      # print("k0")
      # print(abname)
      k0_names = append(k0_names, ko$ko_name[idToAdd][1])
    }
    else{
      k0_names = append(k0_names, ko$ko_name[idToAdd])

    }


  }else{
    # print(paste("Did not in KO:", abname))
    ptrNameSelected = append(ptrNameSelected, abname)
    KOSelected = append(KOSelected, "-")
    k0_names = append(k0_names, "-")
  }
}

# length(k0_names)
# length(KOSelected)
colnames(RNA_fp_data_ko)[c(colstart: ncol(RNA_fp_data))] = k0_names
## Now all genes names were replaced by its KO names, next step is to group them because each KO name
# 2. Group KO
## This is a placeholder to initiate a dataframe with 20 rows, and the first row will be deleted.
new_ko = RNA_fp_data_ko[,10]

## For loop: each unique KO was sum of corresponding genes.
for (k in unique(k0_names)) {
  # k = unique(k0_names)[1]
  to_sum = which(colnames(RNA_fp_data_ko) == k)
  if (length(to_sum) >= 2) {
    RNA_fp_data_ko[, to_sum]
    to_add = rowSums(RNA_fp_data_ko[, to_sum])
  }else{
    to_add = RNA_fp_data_ko[, to_sum]
  }
  new_ko = cbind(new_ko, to_add)


}

# length(unique(k0_names))
```

```r
# ncol(new_ko)
## Delete the first row
new_ko = new_ko[,-1]
## Add KOs to column names
colnames(new_ko) = unique(kO_names)

new_ko = as.data.frame(new_ko)
## Delete "-" because it is sum of genes not mapped to any KO
new_ko_all = new_ko[,-4]
## RNA_fp_data[,5] is the class of methane emission, so rebind to KO data
new_ko_all_methane = cbind(RNA_fp_data[,5],new_ko_all)
new_ko_all_methane = as.data.frame(new_ko_all_methane)
names(new_ko_all_methane)[1] = "Class"



# new_ko_all_methane


new_ko_all_methane_copy = new_ko_all_methane

# 3. WRS test
colstart = 2
pCollection = c()
# write.csv(new_ko_all_methane_copy, "kegg_check.csv")

for (i in c(colstart: ncol(new_ko_all_methane))) {
  # i = 2
  # Not paired
  # x = new_ko_all_methane[,i][new_ko_all_methane$Class == "high"]
  # mean(x)
  # sd(x)
  # #means that it select KO expressions in HMY group
  # y = new_ko_all_methane[,i][new_ko_all_methane$Class == "low"]
  # mean(y)
  # sd(y)
  if (anova_test) {
    paste(colnames(new_ko_all_methane)[i]," ~ ", "Class", sep = "")
    colnames(new_ko_all_methane)[i] = paste("X",i,sep = "")
    test_name = colnames(new_ko_all_methane)[i]
    anove_test = aov(formula = as.formula(paste(test_name,"~ ", "Class", sep = "")),data = new_ko_all
    s_anova = summary(anove_test)
    p_tested = s_anova[[1]][["Pr(>F)"]][1]
  }
  if (wrs_test) {
    wTest = wilcox.test(x = new_ko_all_methane[,i][new_ko_all_methane$Class == "high"],
                        y = new_ko_all_methane[,i][new_ko_all_methane$Class == "low"],
                        paired = F)
    p_tested = wTest$p.value
  }
  if (t_test) {
    tTest = t.test(x = new_ko_all_methane[,i][new_ko_all_methane$Class == "high"],
                   y = new_ko_all_methane[,i][new_ko_all_methane$Class == "low"],
```

```
                    paired = F,
                    var.equal = F)
    p_tested = tTest$p.value
  }

  # Box plot
  # putative aldouronate transport system substrate-binding protein
  # boxplot(new_ko_all_methane$`putative aldouronate transport system substrate-binding protein`~new_

  pCollection = append(pCollection, p_tested)
}
# Temporarily set a alpha just for checking propose
alpha = 0.1
# Make correction by BH. ?p.adjust : The "BH" (aka "fdr") and "BY" methods of Benjamini, Hochberg, an

adjustedP = round(p.adjust(pCollection, method = "BH", n = length(pCollection)),3)
# 4. add mean, sd, and log2FC
## Create a new dataset without first column of new_ko_all_methane
df_kegg = data.frame("KO_name" = colnames(new_ko_all_methane_copy)[-1])



meansC = c()
sdsC = c()
log2_CPM_c = c()
library(stringr)
for (i in c(1:length(df_kegg$KO_name))) {
  # print(i+4)
  # i = 1
  # i+1 because we skipped the first column
  LMY_ptrs = new_ko_all_methane[,i+1][RNA_fp_data$Class == "low"]
  HMY_ptrs = new_ko_all_methane[,i+1][RNA_fp_data$Class == "high"]
  # Means Sds and LogFC
  meanLMY = round(mean(LMY_ptrs),3)
  sdLMY = round(sd(LMY_ptrs),3)
  meanHMY = round(mean(HMY_ptrs),3)
  sdHMY = round(sd(HMY_ptrs),3)
  log2_CPM = round(log2((meanLMY+1)/(meanHMY+1)),3)
  if (!is.na(log2_CPM) & abs(log2_CPM) != Inf) {
    log2_CPM_c = append(log2_CPM_c, log2_CPM)
  }else{
    log2_CPM_c = append(log2_CPM_c, NA)
  }

  # make them as strings
  meanHMY_LMY = paste(meanHMY, "(", sdHMY, ")",sep = "")
  meansC = append(meansC, meanHMY_LMY)
  sdHMY_LMY = paste(meanLMY, "(", sdLMY, ")",sep = "")
  sdsC = append(sdsC, sdHMY_LMY)

}
# length(df_kegg$KO)
```

```
  # length(meansC)

  ## add to the dataset created
  df_kegg$HMY = meansC
  df_kegg$LMY = sdsC
  df_kegg$adjustedP = round(adjustedP, 3)
  df_kegg$LOGFC = log2_CPM_c
  # reorder based on p value and LogFC
  df_kegg = df_kegg[with(df_kegg, order(adjustedP, LOGFC)), ]
  # df_kegg
  # write.csv(df_kegg[with(df_kegg, order(adjustedP, LOGFC)), ], "kegg.csv")
  # new_ko_all_methane
  # write.csv(new_ko_all_methane, "kegg_2.csv")
  # head(new_ko_all_methane)
  return(df_kegg)
}
```

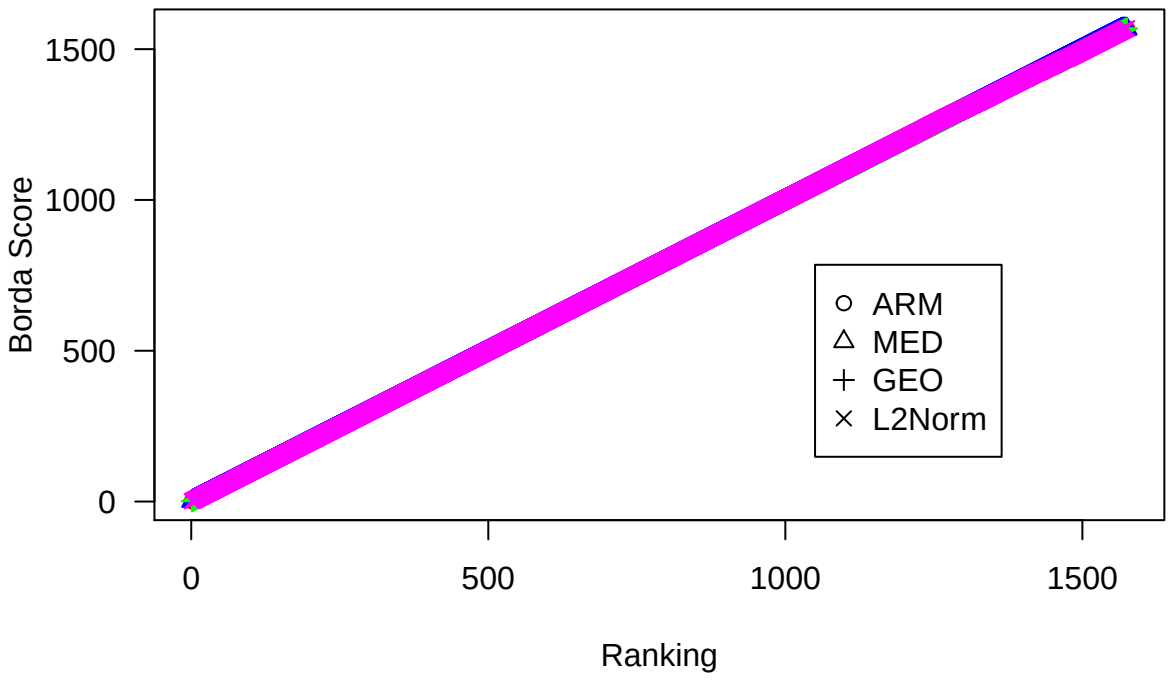
1 IMG gene database

```
RNA_fp_data = read.csv(file = 'RNA_methane_fp.csv', header = T)
```

```
RNA_fp_data = read.csv(file = 'RNA_methane_fp.csv', header = T)
wrs_IMG = df_IMG_tested(RNA_fp_data, wrs_test = T, t_test = F, anova_test = F)
# write.csv(wrs_IMG[with(wrs_IMG, order(adjustedP, LOGFC)), ],"wrs_IMG.csv")
t_IMG = df_IMG_tested(RNA_fp_data, wrs_test = F, t_test = T, anova_test = F)
# write.csv(t_df,"t_df.csv")
anova_IMG = df_IMG_tested(RNA_fp_data, wrs_test = F, t_test = F, anova_test = T)
```

1.1 Borda

```
library(TopKLists)
input=list(wrs_IMG$IMG,t_IMG$IMG,anova_IMG$IMG)
borda_topk=Borda(input)
selection_method = "median"
test_method = "borda"
Borda.plot(borda_topk)
```




```
data.frame(WRS = wrs_IMG$IMG[c(1:10)], borda = borda_topk[["TopK"]][["median"]][1:10])
```

```
##           WRS      borda
## 1  2815273058 2815273058
## 2  2815273063 2815273063
## 3  2815273069 2815273069
## 4  2815273083 2815273083
## 5  2815273088 2815273088
## 6  2815273089 2815273089
## 7  2815273106 2815273106
## 8  2815273107 2815273107
## 9  2815273116 2815273116
## 10 2815273119 2815273119
```

```
library(ggvenn)
```

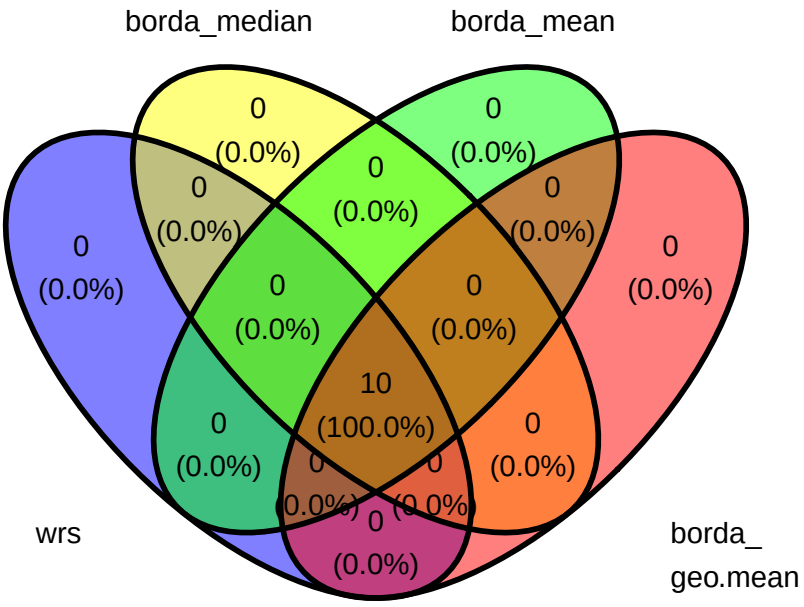
```
## Loading required package: dplyr
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

## Loading required package: grid

## Loading required package: ggplot2
```

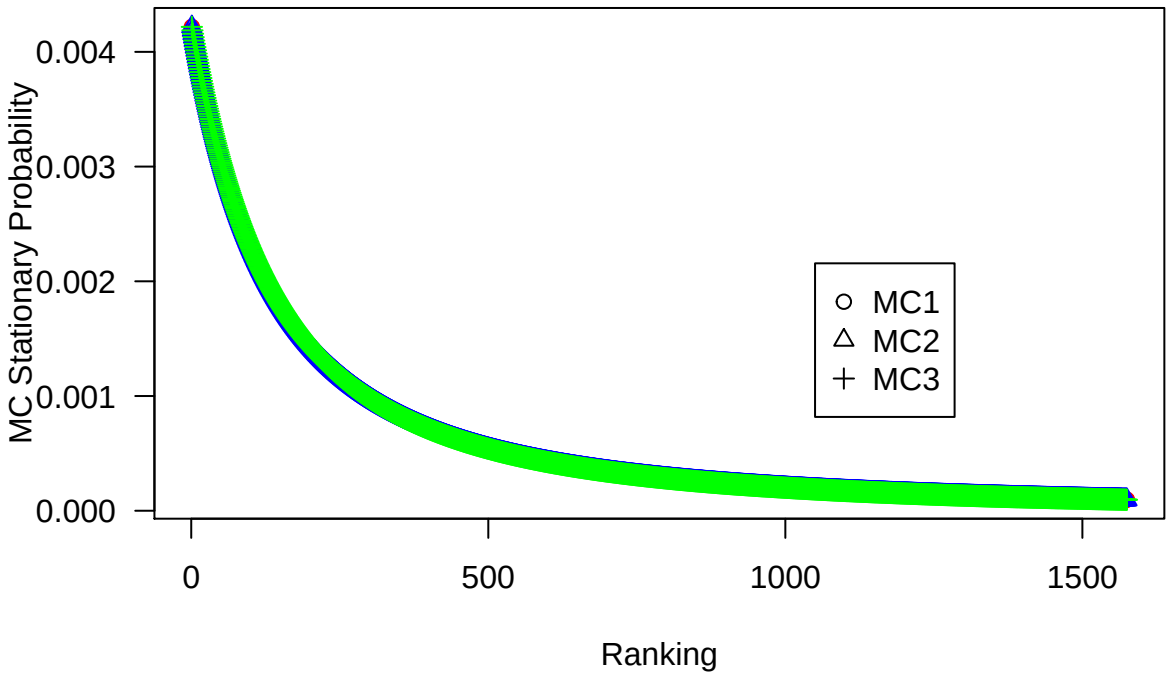
```
library(stringr)
wrs_unique_element = setdiff(wrs_IMG$IMG[c(1:10)], borda_topk[["TopK"]][["median"]][1:10])
borda_unique_element = setdiff(borda_topk[["TopK"]][["median"]][1:10],wrs_IMG$IMG[c(1:10)])
borda_mean_unique_element = setdiff(borda_topk[["TopK"]][["mean"]][1:10],wrs_IMG$IMG[c(1:10)])

ggvenn(set_name_size = 4,list("wrs" = wrs_IMG$IMG[c(1:10)],
                              "borda_median" = borda_topk[["TopK"]][["median"]][1:10],
                              "borda_mean" = borda_topk[["TopK"]][["mean"]][1:10],
                              "borda_\ngeo.mean" = borda_topk[["TopK"]][["geo.mean"]][1:10],
                              "borda_l2norm" = borda_topk[["TopK"]][["l2norm"]][1:10]))
```



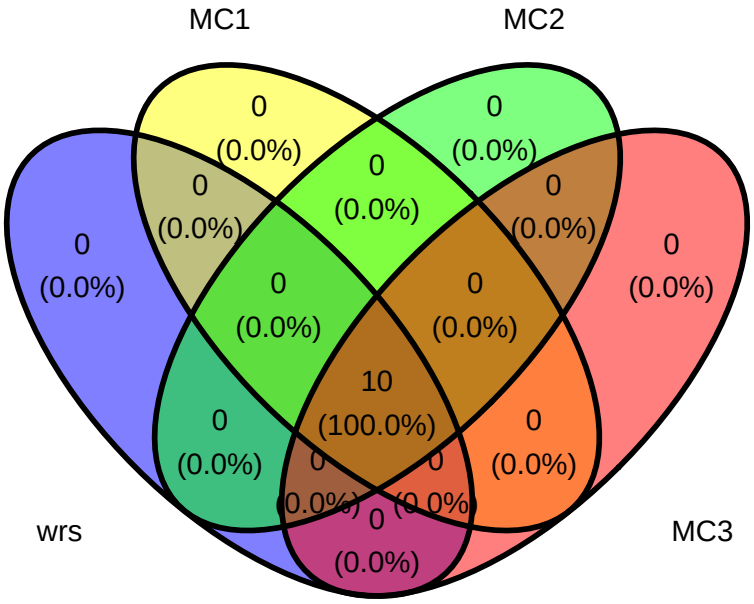
1.2 MC

```
library(TopKLists)
input=list(wrs_IMG$IMG,t_IMG$IMG,anova_IMG$IMG)
```

```
MC_topk=MC(input)
test_method = "MC"
MC.plot(MC_topk)
```




```
library(ggvenn)
library(stringr)


ggvenn(set_name_size = 4,list("wrs" = wrs_IMG$IMG[c(1:10)],
                              "MC1" = MC_topk[["MC1.TopK"]][1:10],
                              "MC2" = MC_topk[["MC2.TopK"]][1:10],
                              "MC3" = MC_topk[["MC3.TopK"]][1:10]))
```

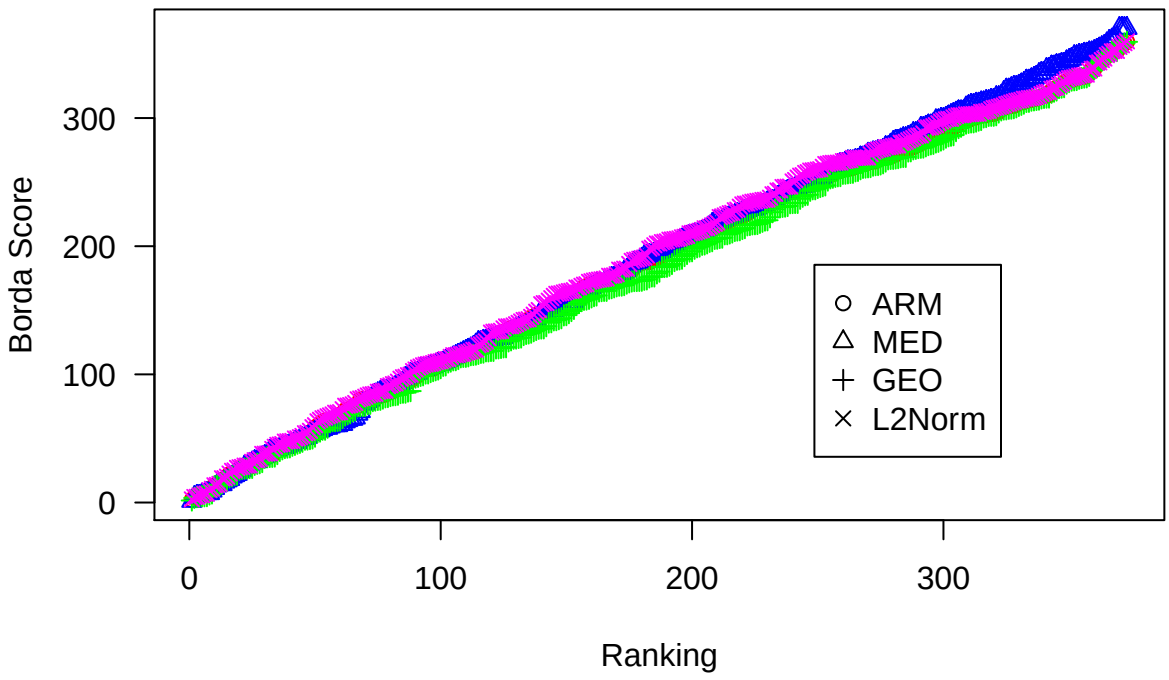


2 KEGG database

```
RNA_fp_data = read.csv(file = 'RNA_methane_fp.csv', header = T)

wrs_df = df_kegg_tested(RNA_fp_data, wrs_test = T, t_test = F, anova_test = F)
# write.csv(wrs_df,"wrs_df.csv")
t_df = df_kegg_tested(RNA_fp_data, wrs_test = F, t_test = T, anova_test = F)
# write.csv(t_df,"t_df.csv")

anova_df = df_kegg_tested(RNA_fp_data, wrs_test = F, t_test = F, anova_test = T)
# write.csv(anova_df,"anova_df.csv")
```

2.1 Borda

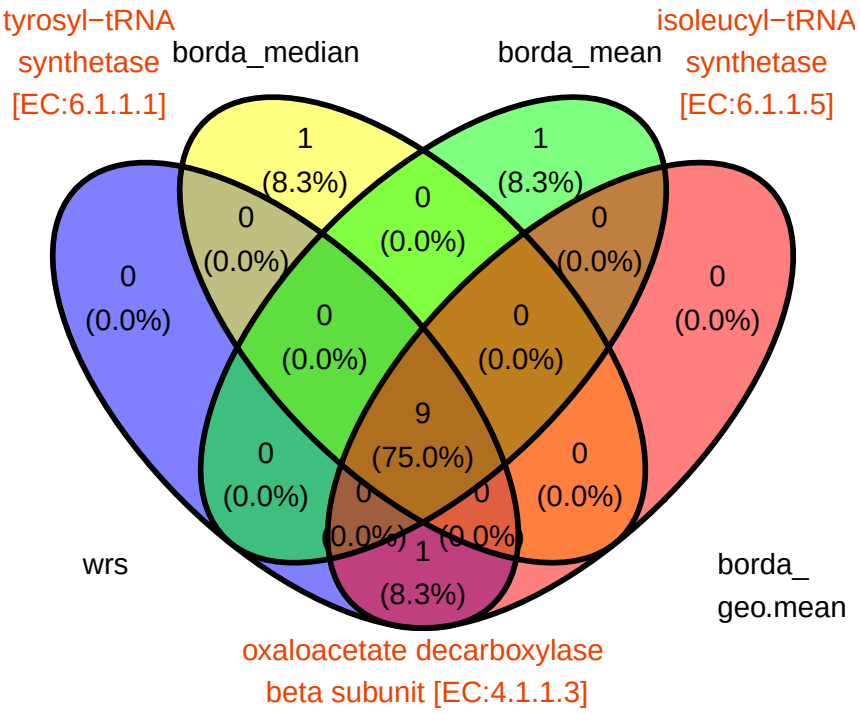
```
library(TopKLists)
input=list(wrs_df$KO_name,t_df$KO_name,anova_df$KO_name)
borda_topk=Borda(input)
selection_method = "median"
test_method = "borda"
Borda.plot(borda_topk)
```




```
data.frame(WRS = wrs_df$KO_name[c(1:10)], borda = borda_topk[["TopK"]][["median"]][1:10])
```

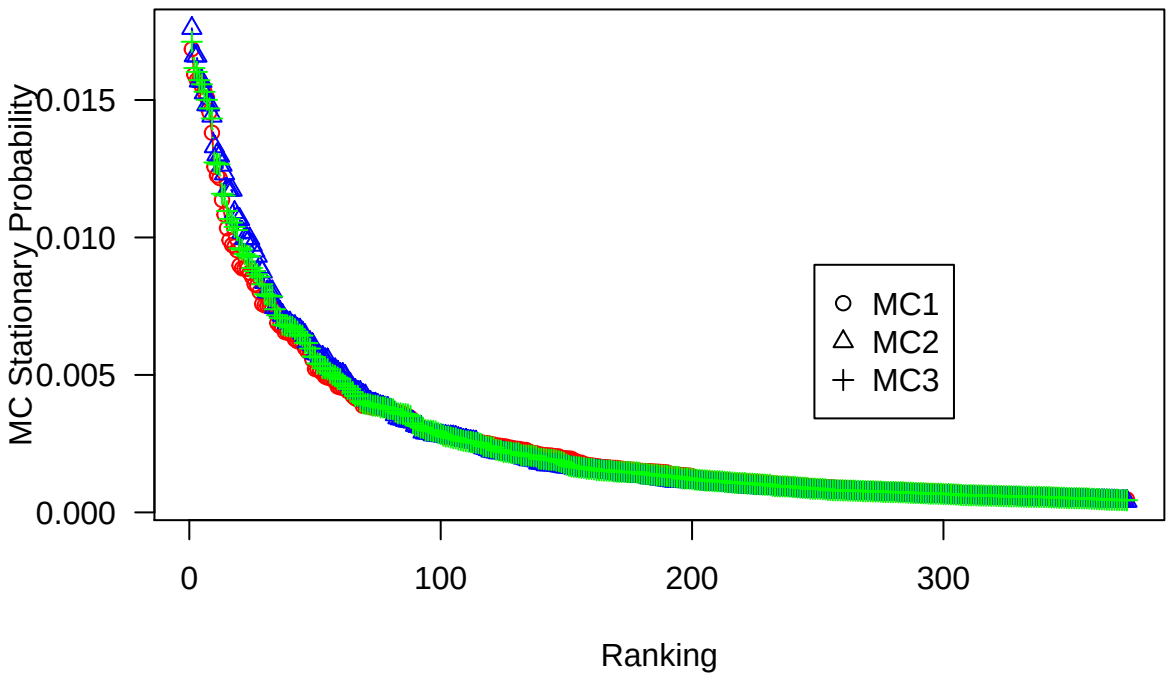
```
##                                                                  WRS
## 1  putative aldouronate transport system substrate-binding protein
## 2           putative aldouronate transport system permease protein
## 3                        formate C-acetyltransferase [EC:2.3.1.54]
## 4  raffinose/stachyose/melibiose transport system permease protein
## 5         PTS system, N-acetylgalactosamine-specific IID component
## 6  simple sugar transport system ATP-binding protein [EC:3.6.3.17]
## 7                              enoyl-CoA hydratase [EC:4.2.1.17]
## 8            oxaloacetate decarboxylase, beta subunit [EC:4.1.1.3]
## 9                          aspartate--ammonia ligase [EC:6.3.1.1]
## 10                      diamine N-acetyltransferase [EC:2.3.1.57]
##                                                               borda
## 1  raffinose/stachyose/melibiose transport system permease protein
## 2  simple sugar transport system ATP-binding protein [EC:3.6.3.17]
## 3                        formate C-acetyltransferase [EC:2.3.1.54]
## 4         PTS system, N-acetylgalactosamine-specific IID component
## 5           putative aldouronate transport system permease protein
## 6                       diamine N-acetyltransferase [EC:2.3.1.57]
## 7                              enoyl-CoA hydratase [EC:4.2.1.17]
## 8                          aspartate--ammonia ligase [EC:6.3.1.1]
## 9  putative aldouronate transport system substrate-binding protein
## 10                          tyrosyl-tRNA synthetase [EC:6.1.1.1]
```

```
library(ggvenn)
library(stringr)
wrs_unique_element = setdiff(wrs_df$KO_name[c(1:10)], borda_topk[["TopK"]][["median"]][1:10])
borda_unique_element = setdiff(borda_topk[["TopK"]][["median"]][1:10],wrs_df$KO_name[c(1:10)])
borda_mean_unique_element = setdiff(borda_topk[["TopK"]][["mean"]][1:10],wrs_df$KO_name[c(1:10)])

ggvenn(set_name_size = 4,list("wrs" = wrs_df$KO_name[c(1:10)],
                              "borda_median" = borda_topk[["TopK"]][["median"]][1:10],
                              "borda_mean" = borda_topk[["TopK"]][["mean"]][1:10],
                              "borda_\ngeo.mean" = borda_topk[["TopK"]][["geo.mean"]][1:10],
                              "borda_l2norm" = borda_topk[["TopK"]][["l2norm"]][1:10]))+
  annotate("text", x=-0, y=-1.9, label= paste(str_split(wrs_unique_element, ",")[[1]],sep = "",collapse
  annotate("text", x=-1.7, y=1.2, label= paste(str_split(borda_unique_element, " ")[[1]],sep = "",colla
  annotate("text", x=1.7, y=1.2, label= paste(str_split(borda_mean_unique_element, " ")[[1]],sep = "",c
```

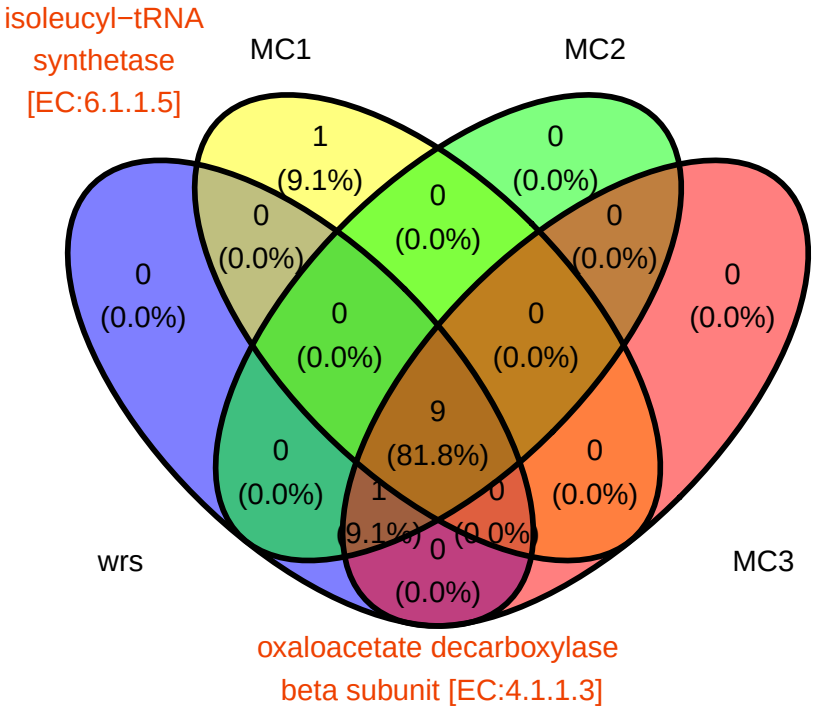


2.2 MC

```
library(TopKLists)
input=list(wrs_df$KO_name,t_df$KO_name,anova_df$KO_name)
MC_topk=MC(input)
test_method = "MC"
MC.plot(MC_topk)
```




```
library(ggvenn)
library(stringr)
wrs_unique_element = setdiff(wrs_df$KO_name[c(1:10)], MC_topk[["MC1.TopK"]][1:10])
# wrs_unique_element
MC_unique_element = setdiff(MC_topk[["MC1.TopK"]][1:10],wrs_df$KO_name[c(1:10)])
borda_mean_unique_element = setdiff(MC_topk[["MC2.TopK"]][1:10],wrs_df$KO_name[c(1:10)])

ggvenn(set_name_size = 4,list("wrs" = wrs_df$KO_name[c(1:10)],
                              "MC1" = MC_topk[["MC1.TopK"]][1:10],
                              "MC2" = MC_topk[["MC2.TopK"]][1:10],
                              "MC3" = MC_topk[["MC3.TopK"]][1:10]))+annotate("text", x=-0, y=-1.9, label
  annotate("text", x=-1.7, y=1.2, label= paste(str_split(MC_unique_element, " ")[[1]],sep = "",collapse
```


isoleucyl−tRNA
synthetase
[EC:6.1.1.5]
MC1
MC2
1
(9.1%)
0
(0.0%)
0
(0.0%)
0
(0.0%)
0
(0.0%)
0
(0.0%)
0
(0.0%)
0
(0.0%)
0
(0.0%)
9
(81.8%)
0
(0.0%)
0
(0.0%)
1
(9.1%)
0
(0.0%)
0
(0.0%)
wrs
MC3
oxaloacetate decarboxylase
beta subunit [EC:4.1.1.3]